

IN THE CLAIMS

Claims 1-17 are canceled herein. Claims 18-37 are pending in the application. Claims 34-38 have been renumbered as claims 33-37 in accordance with the Examiner's request in paragraph 3 of the December 4, 2003 office action. All pending claims are reproduced below.

- 1 1. (Canceled)
- 1 2. (Canceled)
- 1 3. (Canceled)
- 1 4. (Canceled)
- 1 5. (Canceled)
- 1 6. (Canceled)
- 1 7. (Canceled)
- 1 8. (Canceled)
- 1 9. (Canceled)
- 1 10. (Canceled)
- 1 11. (Canceled)

1 12. (Canceled)

1 13. (Canceled)

1 14. (Canceled)

1 15. (Canceled)

1 16. (Canceled)

1 17. (Canceled)

1 18. (Previously Presented) A method for compiling a functional description
2 expressed in an interpretive, algorithmic language into target code for selected hardware, the
3 method comprising the steps of:

4 receiving the functional description expressed in the interpretive, algorithmic
5 language with at least one undeclared variable;

6 assigning a type and a dimension to the at least one undeclared variable by
7 analyzing the functional description to form an abstract syntax tree;

8 transforming compound statements in the abstract syntax tree into a series of
9 single statements; and

10 translating the abstract syntax tree into a register transfer level format.

1 19. (Previously Presented) The method for compiling a functional description
2 of claim 18, further comprising the steps of:

3 receiving a user directive file including at least one user defined directive selected
4 from the group consisting of constraint directives, assertions, and compiler hints; and
5 annotating the functional description according to the user directive file.

1 20. (Previously Presented) The method for compiling a functional description
2 of claim 18, further comprising the steps of:

3 analyzing a value range of the at least one undeclared variable; and
4 assigning a required precision for the at least one undeclared variable.

1 21. (Previously Presented) The method for compiling a functional description
2 of claim 20, further comprising the step of:

3 parsing a real undeclared variable into an integer part and a fractional part,
4 wherein said real undeclared variable is one of said at least one undeclared variable.

1 22. (Previously Presented) The method for compiling a functional description
2 of claim 18, further comprising the steps of :

3 analyzing array access patterns across loop iterations; and

4 replacing a statement in a loop including a memory access with multiple
5 statements including the memory access to reduce the number of individual memory
6 accesses.

1 23. (Previously Presented) The method for compiling a functional description
2 of claim 18, further comprising the steps of:

3 analyzing compound loop structures to identify pipeline opportunities; and
4 applying the pipeline algorithm to pipeline opportunities to generate nodes
5 corresponding to the loop body, predicate nodes corresponding to loop conditional
6 statements, and a schedule for scheduling pipeline operations.

1 24. (Previously Presented) The method for compiling a functional description of
2 claim 18, wherein the step of transforming compound statements in the abstract syntax tree into a
3 series of single statements comprises the step of:

4 expanding a matrix operation into at least one loop.

1 25. (Previously Presented) The method for compiling a functional description of
2 claim 18, wherein the step of transforming compound statements in the abstract syntax tree into a
3 series of single statements comprises the step of:

4 deconstructing a compound statement into at least one simple statement.

1 26. (Previously Presented) A system for compiling a functional description expressed
2 in an interpretive, algorithmic language into target code for selected hardware comprising:

3 a parser for receiving the functional description expressed in the interpretive,
4 algorithmic language with at least one undeclared variable;

5 a type-shape analyzer, coupled to the parser, for assigning a type and a dimension
6 to the at least one undeclared variable by analyzing the functional description to form an
7 abstract syntax tree;

8 a statement deconstructor, coupled to the type-shape analyzer, for transforming a
9 compound statement in the abstract syntax tree into at least one simple statement; and

10 a translator, coupled to the statement deconstructor, for translating the abstract
11 syntax tree into a register transfer level format.

1 27. (Previously Presented) The system for compiling a functional description
2 of claim 26, further comprising:

3 a user directive file, coupled to the parser, for annotating the functional
4 description with at least one user defined directive selected from the group consisting of
5 constraint directives, assertions, and compiler hints.

1 28. (Previously Presented) The system for compiling a functional description
2 of claim 26, further comprising:

3 a precision analyzer, coupled to the type-shape analyzer, for determining the
4 precision of the at least one undeclared variable.

1 29. (Previously Presented) The system for compiling a functional description
2 of claim 28, further comprising:

3 a real number parser, coupled to the precision analyzer, for parsing a real number
4 into an integer part and a fractional part.

1 30. (Previously Presented) The system for compiling a functional description
2 of claim 26, further comprising:

3 a memory access optimizer, coupled to the statement deconstructor, for analyzing
4 array access patterns across loop iterations and replacing a statement in a loop including a
5 memory access with multiple statements including the memory access to reduce the
6 number of individual memory accesses.

1 31. (Previously Presented) The system for compiling a functional description
2 of claim 26, further comprising:

3 a pipeline optimizer, coupled to the statement deconstructor, for analyzing
4 compound loop structures to identify pipeline opportunities and applying the pipeline
5 algorithm to pipeline opportunities to generate nodes corresponding to the loop body,
6 predicate nodes corresponding to loop conditional statements, and a schedule for
7 scheduling pipeline operations.

1 32. (Previously Presented) The system for compiling a functional description
2 of claim 26, wherein the statement deconstructor for transforming a compound statement in the
3 abstract syntax tree into at least one simple statement comprises:

4 a scalarizer, coupled to the type-shape analyzer, for expanding a matrix operation
5 into at least one loop.

1 3433. (Currently Amended) One or more computer readable storage devices having
2 computer readable code embodied on said computer readable storage device, said computer
3 readable code for programming one or more computers to perform a method for compiling a
4 functional description expressed in an interpretive, algorithmic language into target code for
5 selected hardware, the method comprising the steps of:

6 receiving the functional description expressed in the interpretive, algorithmic
7 language with at least one undeclared variable;

8 assigning a type and dimension to the at least one undeclared variable by
9 analyzing the functional description to form an abstract syntax tree;

10 transforming compound statements in the abstract syntax tree into a series of
11 single statements; and

12 translating the abstract syntax tree into a register transfer level format.

1 3534. (Currently Amended) One or more computer readable storage devices having
2 computer readable code embodied on said computer readable storage device, said computer
3 readable code for programming one or more computers to perform a method for compiling a
4 functional description of claim 3433, further comprising the step of:

5 receiving a user directive file including at least one user defined directive selected
6 from the group consisting of constraint directives, assertions, and compiler hints; and

7 annotating the functional description according to the user directive file.

1 3635. (Currently Amended) One or more computer readable storage devices having
2 computer readable code embodied on said computer readable storage device, said computer
3 readable code for programming one or more computers to perform a method for compiling a
4 functional description of claim 3433, further comprising the step of:

5 analyzing a value range of the at least one undeclared variable; and

6 assigning a required precision for the at least one undeclared variable.

1 3736. (Currently Amended) One or more computer readable storage devices having
2 computer readable code embodied on said computer readable storage device, said computer
3 readable code for programming one or more computers to perform a method for compiling a
4 functional description of claim 3433, further comprising the step of:

5 analyzing array access patterns across loop iterations; and

6 replacing a statement in a loop with a memory access with multiple statements
7 with the memory access to reduce the number of individual memory accesses.

1 3837. (Currently Amended) One or more computer readable storage devices having
2 computer readable code embodied on said computer readable storage device, said computer
3 readable code for programming one or more computers to perform a method for compiling a
4 functional description of claim 3433, further comprising the step of:

5 analyzing compound loop structures to identify pipeline opportunities; and

6 applying the pipeline algorithm to pipeline opportunities to generate nodes
7 corresponding to the loop body, predicate nodes corresponding to loop conditional
8 statements, and a schedule for scheduling pipeline operations.